Compiler Lec 10 notes

* Oral Exam of DB 6-1 at 10 AM
* Project Discussion ... Compiler

sec 1 and (sec 2 till no. 11) ⟶ 31-12 at 12 PM.

sec 2 from no. 12 and sec 3 ⇒ 2-1 at 1 PM

← بزيادة ال (lexical) فيه حاجات مبتدرش بيعملها (identification)
فبروح لل (Parser) ← بردو ال (Parser) فيه حاجات مبتدرش
بيتعامل معاها فبروح لل (symantic).

← ال (Phases) الأخيرة في ال (front end)

→ Lexical, Parser, semantic → Page 4 on slide.

← مسئول عن تحديد ال (scope) لل (variables).

← في مراحل ال (front end) كنت بعمل (check) للغات الموجودة معانا فقط.

← يعني اللي (Compiler) بيعالجه لسه متكلمناش عنه.

Run-time ← مقصود بيها ال (execution) ← dynamic

Notations

"OS" ← يعمل (memory allocation) عشان يعمل (space) للبرنامج.

← ~~ال (code) يعمل (code)~~ :

Memory

Low Address

← تحديد (space) يتم وضع ال (code) بـ.

code

other space

← يتوضع بـ ال (data) اللي البرنامج محتاجها.

High Address

← شكل افتراضي لشكل ال (memory)

ال (Compiler) مسئول عن الأوركسترا بتاعة تنظيم استخدام ال (data) في البرنامج. ←

هو "روجر فيدرير" بتاع البرنامج

We need

1) our Program more efficient.

2) our Program is executed rapidly.

→ as fast as correct as possible

1) Execution is sequential ⇒ is this always happens?

← يوجد شيء (لنه) (Parallel Computing) ممكن فجأة البرنامج عندنا يشتغل أكثر من (Processor) فيكون ال (Execution not sequential)

→ life time → is a dynamic Concept.

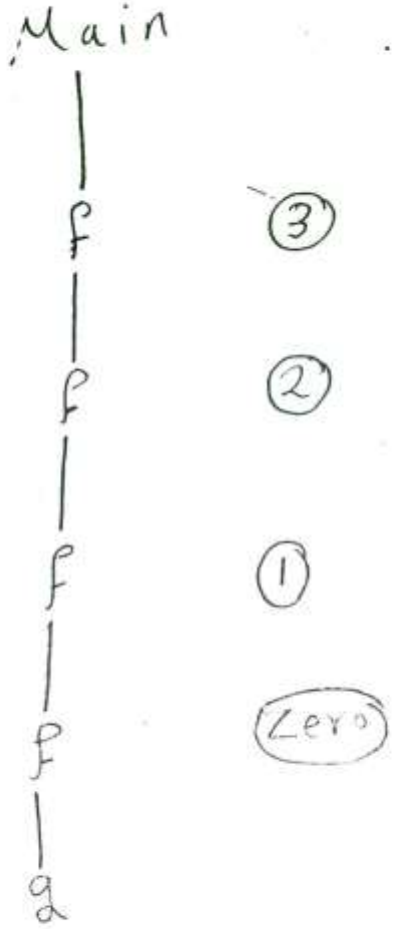
(run-time) يعني (execution) ال وقت بيشتغل ←

scope → static Concept

← ال (scope) هو الجزء اللي أنا شايف فيه ال (Variable)

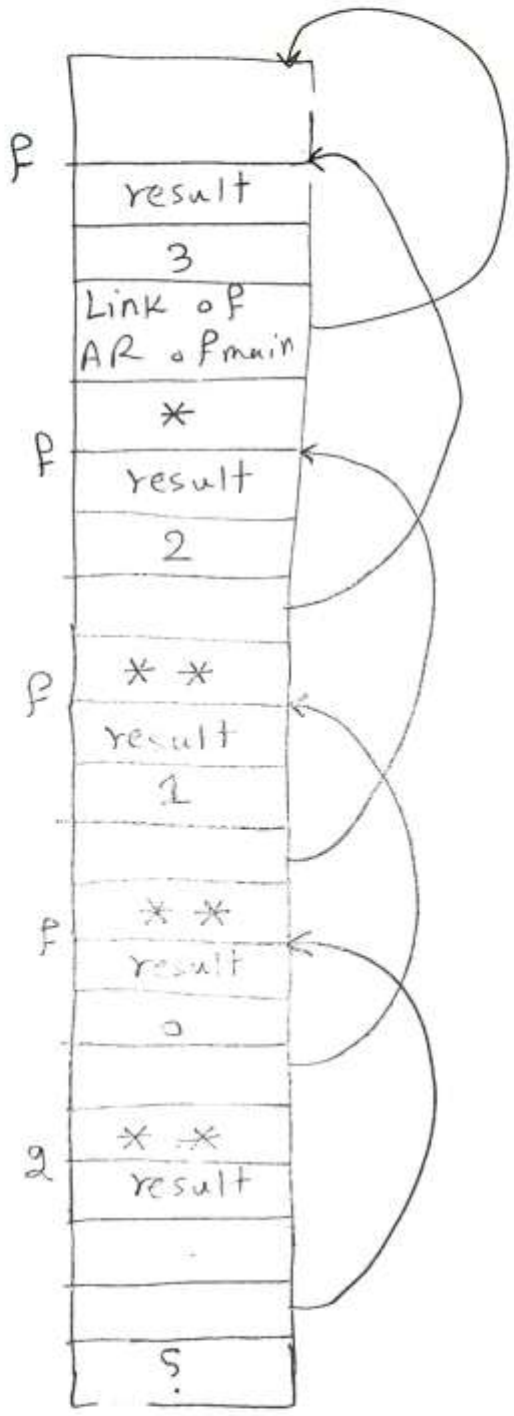
(Compile-time) أنا بعمل ,

Page 21



Stack Part

في الصفحة القادمة



← ممكن تيجي في الإختبار.

ال (offset) ثابت.

← لو وافق عنه أول f ممكن نعمل (SKIP) لـ ٤ مراحل وأوصل لـ f التالية.

Foo => Create new object Called Bar

bar must be accessable by "Foo"

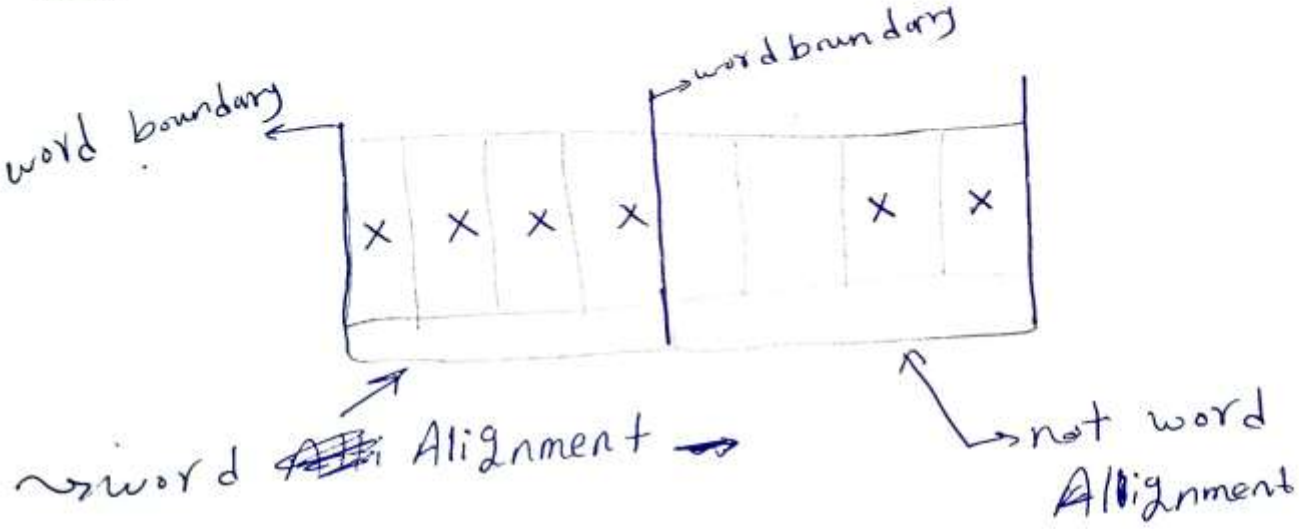
~> Page 42

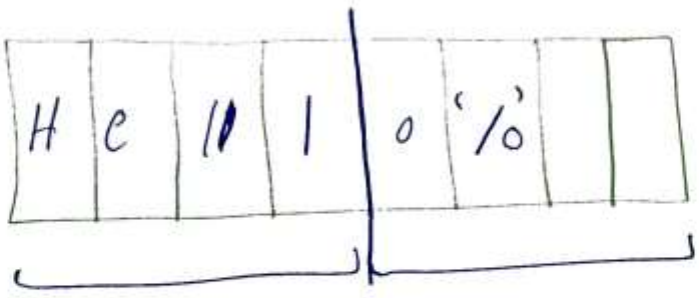
(Heap ، stack) الجزء المحجوز لـ

← لا يحتمل (Crossing) بين

← معناها انه مباشر فيه (memory space) والبرنامج يقف.

(OS) يقوم بحل (Action) لحل المشكلة دي.





← هيعمل (SKIP) للجزء الباقي

stack machine

$$r = F(a_1, a_2, \ldots, a_n)$$

ك operation) تنفذ ها ← عايزة حل ال ← (stack)

PoP ← للي موجود في ال stack

Compute ← للعناوين دي

Push ← (result) ال

(top.f stack) دي هي تاخد

$$op(\underbrace{e_1}_{\text{sub expressions}}, \ldots, e_n)$$

① For each e_i $(0 < i < n)$

Compute e_i ← result in acc

1→n-1 Push result on the stack

e_n ← Just evaluate (no need to Push)

② PoP n-1 values from the stack,
Compute op.

③ store result in Acc.

إنظر لمثالين Page 54